Problem A. AiGo

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Recently, AIs which play Go (a traditional board game) are well investigated. Your friend Hikaru is planning to develop a new awesome Go AI named Sai and promote it to company F or company G in the future. As a first step, Hikaru has decided to develop an AI for 1D-Go, a restricted version of the original Go.

In both of the original Go and 1D-Go, capturing stones is an important strategy. Hikaru asked you to implement one of the functions of capturing.

In 1D-Go, the game board consists of L grids lie in a row. A state of 1D-go is described by a string S of length L. The i-th character of S describes the i-th grid as the following:

- When the i-th character of S is '`B`', the i-th grid contains a stone which is colored black.
- When the i-th character of S is '`W`', the i-th grid contains a stone which is colored white.
- When the i-th character of S is '`.`', the i-th grid is empty.

Maximal continuous stones of the same color are called a chain. When a chain is surrounded by stones with opponent's color, the chain will be captured.

More precisely, if i-th grid and j-th grids ($1 < i + 1 < j \leq L$) contain white stones and every grid of index k ($i < k < j$) contains a black stone, these black stones will be captured, and vice versa about color.

Please note that some of the rules of 1D-Go are quite different from the original Go. Some of the intuition obtained from the original Go may curse cause some mistakes.

You are given a state of 1D-Go that next move will be played by the player with white stones. The player can put a white stone into one of the empty cells. However, the player can not make a chain of white stones which is surrounded by black stones even if it simultaneously makes some chains of black stones be surrounded. It is guaranteed that the given state has at least one grid where the player can put a white stone and there are no chains which are already surrounded.

Write a program that computes the maximum number of black stones which can be captured by the next move of the white stones player.

Input

First line of the input contains one integer L ($1 \leq L \leq 100$) means the length of the game board and S ($|S| = L$) is a string which describes the state of 1D-Go. The given state has at least one grid where the player can put a white stone and there are no chains which are already surrounded.

Output

Output the maximum number of stones which can be captured by the next move in a line.

Examples

standard input	standard output
`5 .WB..`	`1`
`5 .WBB.`	`2`
`6 .WB.B.`	`0`
`6 .WB.WB`	`0`

Note

In the 3rd and 4th test cases, the player cannot put a white stone on the 4th grid since the chain of the white stones will be surrounded by black stones. This rule is different from the original Go.

In the 5th test case, the player cannot capture any black stones even if the player put a white stone on the 4th grid. The player cannot capture black stones by surrounding them with the edge of the game board and the white stone. This rule is also different from the original Go.

Problem B. Non-Trivial Common Divisor

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are given a positive integer sequence A of length N. You can remove any numbers from the sequence to make the sequence "friendly". A sequence is called friendly if there exists an integer k ($k > 1$) such that every number in the sequence is a multiple of k. Since the empty sequence is friendly, it is guaranteed that you can make the initial sequence friendly.

You noticed that there may be multiple ways to make the sequence friendly. So you decide to maximize the sum of all the numbers in the friendly sequence. Please calculate the maximum sum of the all numbers in the friendly sequence which can be obtained from the initial sequence.

Input

The input consists of a single test case formatted as follows: the first line consists of a single integer N ($1 \leq N \leq 1000$). The $i+1$-st line consists of an integer A_i ($1 \leq A_i \leq 10^9$) for ($1 \leq i \leq N$).

Output

Print the maximum sum of all the numbers in the friendly sequence which can be obtained from the initial sequence.

Examples

standard input	standard output
6 1 2 3 4 5 6	12
3 173 1733 111733	111733
4 1 1 1 1	0
10 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999	9999999990
1 999999999	999999999
10 28851 8842 9535 2311 25337 26467 12720 10561 8892 6435	56898

Problem C. Parity Sort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You have a permutation P of length n. In this problem, elements of the permutation are integers from 0 to $n-1$.

Your task is to perform the following operation up to 30 times to sort P in ascending order.

The operation is defined by two parameters: an integer t denoting the mode of operation ($0 \le t \le 1$) and a string S of length n, consisting of `0`s and `1`s.

At the start of the process, we have two empty sequences, A and B.

Next, for each i from 1 to n, we repeat the following step:

- If $S_i = 0$, do nothing.
- If $S_i = 1$: if P_i is even, add P_i to the end of sequence A, otherwise add P_i to the end of sequence B.

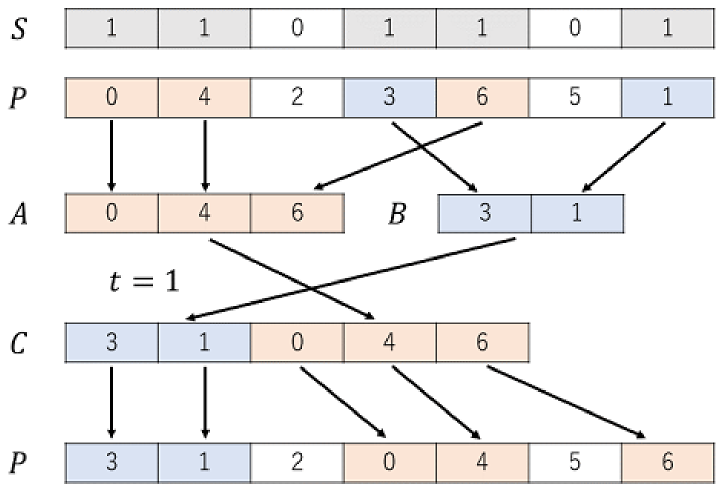
If $t = 0$, sequence C is the concatenation of sequence A and sequence B in that order.

If $t = 1$, sequence C is the concatenation of sequence B and sequence A in that order.

Next, for each i fron 1 to n, we repeat the following step:

- If $S_i = 0$, do nothing.
- If $S_i = 1$, replace P_i with the first element of C and erase the first element of C.

For example, if $n = 7$, $P = \{0, 4, 2, 3, 6, 5, 1\}$ and we choose $t = 1$ and $S =$ `1101101`, the process is shown on the picture below.



Input

The first line of the input contains one integer n ($1 \le n \le 15\,000$). The second line contains n integers P_i ($0 \le P_i \le n-1$, $P_i \ne P_j$ if $i \ne j$).

Output

Print one of the possible sorting sequences in the following format:

On the first line, print one integer k: the number of operations ($0 \le k \le 30$).

The i-th of the following k lines shall describe the i-th operation and contain integer t_i ($0 \le t_i \le 1$) and binary string S of length n.

If there is more than one such sequence, choose any one of them. Note that you don't need to minimize k.

Example

standard input	standard output
7 0 4 2 3 6 5 1	1 1 0100101

Problem D. Permutation Sort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

One day (call it day 0), you find a permutation P of N integers written on the blackboard in a single row. Fortunately you have another permutation Q of N integers, so you decide to play with these permutations.

Every morning of the day 1, 2, 3 you rewrite every number on the blackboard in such a way that erases the number x and write the number Q_x at the same position. Please find the minimum non-negative integer d such that in the evening of the day d the sequence on the blackboard is sorted in increasing order.

Input

The input consists of a single test case in the format below.

The first line of the input contains an integer N ($1 \le N \le 200$). The second line contains N integers $P_1, \ldots, P_N$ ($1 \le P_i \le N$) which represent the permutation P. The third line contains N integers $Q_1, \ldots, Q_N$ ($1 \le Q_i \le N$ which represent the permutation Q.

Output

Print the minimum non-negative integer d such that in the evening of the day d the sequence on the blackboard is sorted in increasing order. If such d does not exist, print -1 instead. It is guaranteed that the answer does not exceed 10^{18}.

Examples

standard input	standard output

6
2 3 1 4 5 6
3 1 2 5 4 6
	4

6
1 2 3 4 5 6
3 1 2 5 4 6
	0

6
2 3 1 4 5 6
3 4 5 6 1 2
	-1

Problem E. Consistent Trading

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Your company is developing a video game. In this game, players can exchange items. This trading follows the rule set by the developers. The rule is defined as the following format: "Players can exchange one item A_i and x_i items B_i". Note that the trading can be done in both directions. Items are exchanged between players and the game system. Therefore players can exchange items any number of times.

Sometimes, testers find bugs that a repetition of a specific sequence of tradings causes the unlimited increment of items. For example, the following rule set can cause this bug.

- Players can exchange one item 1 and two item 2.
- Players can exchange one item 2 and two item 3.
- Players can exchange one item 1 and three item 3.

In this rule set, players can increase items unlimitedly. For example, players start tradings with one item 1. Using rules 1 and 2, they can exchange it for four item 3. And, using rule 3, they can get one item 1 and one item 3. By repeating this sequence, the amount of item 3 increases unlimitedly.

These bugs can spoil the game, therefore the developers decided to introduce the system which prevents the inconsistent trading. Your task is to write a program which detects whether the rule set contains the bug or not.

Input

The input consists of a single test case in the format below.

The first line contains two integers N and M which are the number of types of items and the number of rules, respectively ($1 \le N \le 10^5$, $1 \le M \le 10^5$). Each of the following M lines gives the trading rule that one item A_i and x_i item B_i ($1 \le A_i, B_i \le N$, $1 \le x_i \le 10^9$) can be exchanged in both directions. There are no exchange between same items, i.e., $A_i \ne B_i$.

Output

If there are no bugs, i.e., repetition of any sequence of tradings does not cause an unlimited increment of items, output "`Yes`". If not, output "`No`".

Examples

standard input	standard output

4 4
1 2 2
2 3 2
3 4 2
4 2 3
	No

4 3
1 2 7
2 3 5
4 1 2
	Yes

4 4
1 2 101
2 3 99
1 4 100
4 3 100
	No

5 6
3 1 4
2 3 4
5 4 15
2 1 16
2 4 20
5 3 3
	Yes

Problem F. All your base are belong to us

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In A.D. 2101, war was beginning. The enemy has taken over all of our bases. To recapture the bases, we decided to set up a headquarters. We need to define the location of the headquarters so that all bases are not so far away from the headquarters. Therefore, we decided to choose the location to minimize the sum of the distances from the headquarters to the furthest K bases. The bases are on the 2-D plane, and we can set up the headquarters in any place on this plane even if it is not on a grid point.

Your task is to determine the optimal headquarters location from the given base positions.

Input

The first line of the input contains two integers N and K. The integer N is the number of the bases ($1 \leq N \leq 200$). The integer K gives how many bases are considered for calculation ($1 \leq K \leq N$). Each of the following N lines gives two integers x and y — coordinates of each base. All of the absolute values of given coordinates are less than or equal to 1000, i.e., $-1000 \leq x_i, y_i \leq 1000$ is satisfied.

Output

Output the minimum sum of the distances from the headquarters to the furthest K bases. The output can contain an absolute or a relative error no more than 10^{-3}.

Examples

standard input	standard output
`3 1`	`0.70711`
`0 1`	
`1 0`	
`1 1`	
`6 3`	`17.50426`
`1 1`	
`2 1`	
`3 2`	
`5 3`	
`8 5`	
`13 8`	
`9 3`	`1841.20904`
`573 -50`	
`-256 158`	
`-751 14`	
`314 207`	
`293 567`	
`59 -340`	
`-243 -22`	
`-268 432`	
`-91 -192`	

Problem G. Route Calculator Returns

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You have a grid with H rows and W columns. Each cell contains one of the following 11 characters: an addition operator '`+`', a multiplication operator '`*`', or a digit between `1` and `9`.

There are paths from the top-left cell to the bottom-right cell by moving right or down $H + W - 2$ times. Let us define the value of a path by the evaluation result of the mathematical expression you can obtain by concatenating all the characters contained in the cells on the path in order. Your task is to compute the sum of values of any possible paths. Since the sum can be large, find it modulo M.

It is guaranteed the top-left cell and the bottom-right cell contain digits. Moreover, if two cells share an edge, at least one of them contains a digit. In other words, each expression you can obtain from a path is mathematically valid.

Input

The first line of the input consists of three integers H, W and M ($1 \le H, W \le 2000$, $2 \le M \le 10^9$). The following H lines represent the characters on the grid. $a_{i,j}$ represents the character contained in the cell at the i-th row and j-th column. Each $a_{i,j}$ is either '`+`', '`*`', or a digit between 1 and 9. $a_{1,1}$ and $a_{H,W}$ are both digits. If two cells share an edge, at least one of them contain a digit.

Output

Print the sum of values of all possible paths modulo M.

Examples

standard input	standard output
2 3 1000 3*1 +27	162
4 4 3000000 24+7 *23* 9+48 *123	2159570

Problem H. N-by-M grid calculation

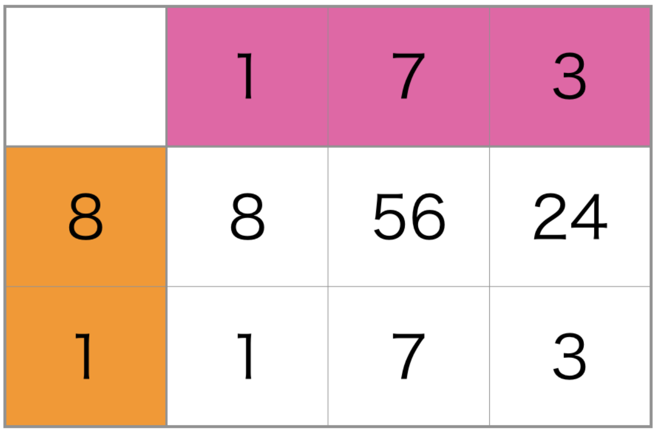
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Have you experienced 1010-by-1010 grid calculation? It's a mathematical exercise common in Japan. In this problem, we consider the generalization of the exercise, N-by-M grid calculation.

In this exercise, you are given an N-by-M grid (i.e. a grid with N rows and M columns) with an additional column and row at the top and the left of the grid, respectively. Each cell of the additional column and row has a positive integer. Let's denote the sequence of integers on the column and row by a and b, and the i-th integer from the top in the column is a_i and the j-th integer from the left in the row is b_j, respectively.

Initially, each cell in the grid (other than the additional column and row) is blank. Let (i, j) be the cell at the i-th from the top and the j-th from the left. The exercise expects you to fill all the cells so that the cell (i, j) has $a_i \times b_j$. You have to start at the top-left cell. You repeat to calculate the multiplication $a_i \times b_j$ for a current cell (i, j), and then move from left to right until you reach the rightmost cell, then move to the leftmost cell of the next row below.

At the end of the exercise, you will write a lot, really a lot of digits on the cells. Your teacher, who gave this exercise to you, looks like bothering to check entire cells on the grid to confirm that you have done this exercise. So the teacher thinks it is OK if you can answer the d-th digit (not integer, see an example below), you have written for randomly chosen x. Let's see an example.



	1	7	3
8	8	56	24
1	1	7	3

For this example, you calculate values on cells, which are 8, 56, 24, 1, 7, 3 in order. Thus, you would write digits 8, 5, 6, 2, 4, 1, 7, 3. So the answer to a question 4 is 2.

You noticed that you can answer such questions even if you haven't completed the given exercise. Given a column a, a row b, and Q integers d_1,d_2, $\ldots$, d_Q, your task is to answer the d_k-th digit you would write if you had completed this exercise on the given grid for each k. Note that your teacher is not so kind (unfortunately), so may ask you numbers greater than you would write. For such questions, you should answer 'x' instead of a digit.

Input

The first line of the input contains two integers N $(1 \leq N \leq 10^5)$ and M $(1 \leq M \leq 10^5)$, which are the number of rows and columns of the grid, respectively.

The second line represents a sequence aa of N integers, the i-th of which is the integer at the i-th from the top of the additional column on the left. It holds $1 \leq a_i \leq 10^9$ for $1 \leq i \leq N$.

The third line represents a sequence b of M integers, the j-th of which is the integer at the j-th from the left of the additional row on the top. It holds $1 \leq b_j \leq 10^9$ for $1 \leq j \leq M$.

The fourth line contains an integer Q $(1 \leq Q \leq 3 \cdot 10^5)$, which is the number of questions your teacher would ask.

The fifth line contains a sequence d of Q integers, the k-th of which is the k-th question from the teacher, and it means you should answer the d_k-th digit you would write in this exercise. It holds $1 \le d_k \le 10^{15}$ for $1 \le k \le Q$.

Output

Output a string with Q characters, the k-th of which is the answer to the k-th question in one line, where the answer to k-th question is the d_k-th digit you would write if d_k is no more than the number of digits you would write, otherwise ‘x’.

Examples

standard input	standard output

2 3
8 1
1 7 3
5
1 2 8 9 1000000000000000
	853xx

3 4
271 828 18
2845 90 45235 3
7
30 71 8 61 28 90 42
	7x406x0

Problem I. Zombie Land

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

Your friend, Tatsumi, is a producer of Immortal Culture Production in Chiba (ICPC). His company is planning to form a zombie rock band named Gray Faces and cheer Chiba Prefecture up.

But, unfortunately, there is only one zombie in ICPC. So, Tatsumi decided to release the zombie on a platform of Soga station to produce a sufficient number of zombies. As you may know, a zombie changes a human into a new zombie by passing by the human. In other words, a human becomes a zombie when the human and a zombie are at the same point. Note that a zombie who used to be a human changes a human into a zombie too.

The platform of Soga station is represented by an infinitely long line, and Tatsumi will release a zombie at a point with coordinate x_Z. After the release, the zombie will start walking in the positive direction at v_Z per second. If v_Z is negative, the zombie will walk in the negative direction at $|v_Z|$ per second.

There are N humans on the platform. When Tatsumi releases the zombie, the i-th human will be at a point with coordinate xi and will start walking in the positive direction at vi per second. If v_i is negative, the human will walk in the negative direction at $|vi|$ per second as well as the zombie.

For each human on the platform, Tatsumi wants to know when the human becomes a zombie. Please help him by writing a program that calculates a time when each human on the platform becomes a zombie.

Input

The first line of the input consists of an integer N ($1 \le N \le 2 \cdot 10^5$) which is the number of humans on a platform of Soga station. The second line consists of two integers x_Z ($-10^9 \le x_Z \le 10^9$) and v_Z($-10^9 \le v_Z \le 10^9$) separated by a space, where x_Z is an initial position of a zombie Tatsumi will release and v_Z is the velocity of the zombie.

The i-th line in the following N lines contains two integers x_i ($-10^9 \le x_i \le 10^9$) and v_i ($-10^9 \le v_i \le 10^9$) separated by a space, where the x_i is an initial position of the i-th human and v_i is the velocity of the human. There is no human that shares their initial position with the zombie. In addition, initial positions of the humans are different from each other.

Output

Print N lines. In the i-th line, print how many seconds it will take for the i-th human to become a zombie. If the i-th human will never become a zombie, print 1 instead. The answer will be considered as correct if the values output have an absolute or relative error less than 10^{-9}.

Examples

standard input	standard output

6
3 1
-5 0
5 0
-4 -3
0 -2
6 -3
2 -1
	3.66666666666667
	2.00000000000000
	-1
	6.00000000000000
	0.75000000000000
	2.00000000000000

5
31415 -926
5358 979
323846 26
-433832 7950
288 -4
-1971 -69
	13.67821522309711
	95.61812216052499
	52.41629112212708
	33.76030368763558
	38.95682613768962

Problem J. Rooks Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Rooks Game is a single-player game and uses a chessboard which has $N \times N$ grid and M rook pieces.

A rook moves through any number of unoccupied squares horizontally or vertically. When a rook can attack another rook, it can capture the rook and move to the square which was occupied. Note that, in Rooks Game, we don't distinguish between white and black, in other words, every rook can capture any of other rooks.

Initially, there are M rooks on the board. In each move, a rook captures another rook. The player repeats captures until any rook cannot be captured. There are two types of goal of this game. One is to minimize the number of captured rooks, and the other is to maximize it.

In this problem, you are requested to calculate the minimum and maximum values of the number of captured rooks.

Input

The first line contains two integers N and M which are the size of the chessboard and the number of rooks, respectively ($1 \le N, M \le 1000$). Each of the following M lines gives the position of each rook. The i-th line with x_i and y_i means that the i-th rook is in the x_i-th column and y_i-th row ($1 \le x_i, y_i \le N$). You can assume any two rooks are not in the same place.

Output

Output the minimum and maximum values of the number of captured rooks separated by a single space.

Examples

standard input	standard output

8 3
1 1
1 8
8 8
	1 2

8 4
1 1
1 8
8 8
8 1
	2 3

5 5
1 1
2 2
3 3
4 4
5 5
	0 0

100 1 100 100	0 0

10 12
1 2
1 4
1 6
3 6
5 6
10 7
8 7
6 7
4 7
2 7
7 3
9 5
	7 8

Problem K. Bombing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

JAG land is a country, which is represented as an $M \times M$ grid. Its top-left cell is $(1, 1)$ and its bottom-right cell is (M, M).

Suddenly, a bomber invaded JAG land and dropped bombs to the country. Its bombing pattern is always fixed and represented by an $N \times N$ grid. Each symbol in the bombing pattern is either '`X`' (bomb) or '`.`' (empty).

Here, suppose that a bomber is in (b_r, b_c) in the land and drops a bomb. The cell $(b_r + i - 1, b_c + j - 1)$ will be damaged if the symbol in the i-th row and the j-th column of the bombing pattern is '`X`' $(1 \leq i, j \leq N)$.

Initially, the bomber reached $(1, 1)$ in JAG land. The bomber repeated to move to either of 4-directions and then dropped a bomb just L times. During this attack, the values of the coordinates of the bomber were between 1 and $MN + 1$, inclusive, while it dropped bombs. Finally, the bomber left the country.

The moving pattern of the bomber is described as L characters. The i-th character corresponds to the i-th move and the meaning of each character is as follows.

'`U`' — up, '`D`' — down, '`L`' — left and '`R`' — right.

Your task is to write a program to analyze the damage situation in JAG land. To investigate damage overview in the land, calculate the number of cells which were damaged by the bomber at least K times.

Input

The first line of the input contains four integers N, M, K and L $(1 \leq N \leq M \leq 500, 1 \leq K \leq L \leq 2 \cdot 10^5)$. The following N lines represent the bombing pattern. B_i is a string of length N. Each character of B_i is either '`X` or '`.`'. The last line denotes the moving pattern. S is a string of length L, which consists of either '`U`', '`D`', '`L`' or '`R`'. It's guaranteed that the values of the coordinates of the bomber are between 1 and $MN + 1$, inclusive, while it drops bombs in the country.

Output

Print the number of cells which were damaged by the bomber at least K times.

Examples

standard input	standard output
2 3 2 4 XX X. RDLU	3
8 10 1 3 XXX.XX.. .XX...X. XX.XXXXX .... XXX.X..X .X.XX..X ..X.X.X. X.XX..X. RRD	63

Problem L. The Spellbook

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You have a spellbook with n spells. The spells are numbered by sequential integers from 1 to n, and the spell i $(1 \le i \le n)$ initially costs A_i MP (mana points). Initially you have m MP.

Your goal is to cast each spell from the spellbook exactly once.

Before you start casting spells, you can eat up to k cookies. Eating a cookie takes zero time. Each time you eat a cookie, you can choose a spell with a positive cost and reduce that cost by 1.

After eating cookies, you start casting spells.

You can repeatedly choose and perform one of the following actions:

- Choose an integer i $(1 \le i \le n)$ and cast the spell i. However, the current MP must be greater than or equal to A_i. This action takes zero time and decreases your MP by A_i.
- If your MP is $z < m$, you may take a rest to restore 1 MP. It takes $m - z$ seconds (for example, if $m = 5$ and $z = 2$, you need to rest for 3 seconds to regenerate MP from 2 to 3).

Find the minimum amount of time you can spend to cast each of the n spells exactly once. You are free to select the order of the spells.

Input

First line of the input contains three integers n, m and k: the number of spells, the initial MP value and the number of cookies, respectively. The second line contains n integers A_i: the initial costs of the spells in MP ($1 \le n \le 10^5$, $1 \le m \le 10^6$, $1 \le A_i \le m$, $0 \le k \le \sum_{i=1}^{n} A_i$).

Output

Print one integer: the minimum amount of time it takes to cast each of the n spells exactly once.

Examples

standard input	standard output
2 4 0 2 4	3
3 9 6 2 3 9	0
3 16 2 6 9 9	21
2 1000000 0 1000000 1000000	500000500000

Problem M. Universal and Existential Quantifiers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a list of N intervals. The i-th interval is $[l_i, r_i)$, which denotes a range of numbers greater than or equal to l_i and strictly less than r_i. In this task, you consider the following two numbers:

- The minimum integer x such that you can select x intervals from the given N intervals so that the union of the selected intervals is $[0, L)$.
- The minimum integer y such that for all possible combinations of y intervals from the given N interval, it does cover $[0, L)$.

We ask you to write a program to compute these two numbers.

Input

The input consists of a single test case formatted as follows.

The first line contains two integers N ($1 \le N \le 2 \cdot 10^5$) and L ($1 \le L \le 10^{12}$), where N is the number of intervals and L is the length of range to be covered, respectively. The i-th of the following N lines contains two integers l_i and r_i ($0 \le l_i < r_i \le L$), representing the range of the i-th interval $[l_i, r_i)$. You can assume that the union of all the N intervals is $[0, L)$.

Output

Output two integers x and y mentioned in the problem statement, separated by a single space, in a line.

Examples

standard input	standard output

3 3
0 2
1 3
1 2
	2 3

2 4
0 4
0 4
	1 1

5 4
0 2
2 4
0 3
1 3
3 4
	2 4

Problem N. Cactus

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

A *cactus graph* is a connected undirected graph without self-loops and multiple edges in which each edge belongs to at most one simple cycle.

Given is a cactus graph G with N vertices, numbered from 1 to N, and M edges. The i-th edge connects vertices a_i and b_i, and its cost is c_i.

Let the cost of a simple path on graph G be bitwise XOR of the costs of all the edges on that path.

Answer Q queries of the form "x_i y_i k_i": consider the costs of all simple paths connecting vertices x_i and y_i, remove duplicate values, sort the values in ascending order, and take k_i-th element. If the number of these values is less than k_i, answer -1.

Input

The first line of the input contains two integers N and M: the number of vertices and the number of edges in the graph ($2 \le N \le 10^5$, $N - 1 \le M \le 2 \cdot 10^5$).

Each of the next M lines describes one edge and contains three integers a_i, b_i and c_i ($1 \le a_i, b_i \le N$, $a_i \ne b_i$, $0 \le c_i < 2^{30}$).

Then follows a line containing an integer Q: the number of queries ($1 \le q \le 2 \cdot 10^5$).

Each of the next Q lines describes one query and contains three integers x_i, y_i and k_i ($1 \le x_i, y_i \le N$, $x_i \ne y_i$, $1 \le k_i \le 2^{30}$).

It is guaranteed that the graph given in the input is a cactus graph.

Output

For each query, print one integer on a separate line: the answer to that query.

Example

standard input	standard output

4 4
1 2 2
1 3 8
2 3 0
1 4 6
4
2 1 1
1 2 2
4 1 1
4 3 2020
	2
	8
	6
	-1

Problem O. Towns and Roads

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There are n towns and $n-1$ bidirectional roads. Towns are numbered from 1 to n, and roads are numbered from 1 to $n-1$, respectively. Road i connects town a_i and town b_i and has length d_i.

At the beginning, all roads are open, and from any town, you can reach all other towns using one or more roads.

There is a robot, initially in town 1.

Your task is to process q queries. Each query has one of the following three types:

- `1` x: move robot to town x. At the time of this query, it is guaranteed that the town where robot is located and town x are directly connected by one open road.
- `2` y: road y is closed. At the time of this query, it is guaranteed that road y is open.
- `3` z: road z is opened again. At the time of this query, it is guaranteed that road z is closed.

In addition, immediately after each query, print the list of towns that are farthest from the town where the robot currently is, if we consider only roads that are open after this query.

Input

The first line of the input contains one integer n, the number of towns ($1 \le n \le 2 \cdot 10^5$).

The i-th of the following $n-1$ lines contains three integers a_i, b_i and d_i: the numbers of cities connected by i-th road and the length of this road, respectively ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le d_i \le 10^6$). It is guaranteed that, from any town, you can reach all other towns using one or more roads.

The next line contains one integer q ($1 \le q \le 2 \cdot 10^5$).

Then q queries follow. Each query is given on a separate line and has one of the three forms described above.

Output

Print q lines: the i-th line should contain the answer to the i-th query.

For each query, start the line with an integer c_i: the number of towns that are farthest from the town where the robot is after the i-th query. Then print c_i integers on the same line: the numbers of these towns in ascending order.

It is guaranteed that, in each test given to your solution, the sum of all c_i in the correct answer will not exceed $4 \cdot 10^5$.

Examples

standard input	standard output

6
2 4 1
1 2 1
4 6 1
2 3 1
4 5 1
5
2 5
2 3
1 2
3 5
1 4
	1 6
	2 3 4
	3 1 3 4
	1 5
	2 1 3

5
3 4 1
2 1 1
4 5 1
3 2 1
6
2 2
3 2
1 2
1 3
2 4
1 4
	1 1
	1 5
	1 5
	2 1 5
	1 5
	2 3 5